Санкт-Петербургский Государственный Университет
Кафедра Компьютерных Технологий и Систем

Зарубин Александр Витальевич

Выпускная квалификационная работа бакалавра

Применение генетического алгоритма для поиска контура объекта

Направление 010300
Фундаментальные информатика и информационные технологии

Научный руководитель,
доктор физ.-мат. наук,
профессор
Веремей Е.И.

Санкт-Петербург
2016

Содержание

Введение

В последние годы влияние технологий на нашу жизнь значительно возросло: до недавнего времени сложно было представить, что человек сможет носить у себя в кармане компактное электронное устройство, сравнимое по производительности с суперкомпьютерами двадцатого века, на которых решались сложнейшие для их эпохи задачи и проводились научные исследования. В наше время, имея такую мощность в кармане, люди не спешат исследовать вселенную или решать насущные задачи человечества.

Данные компактные устройства, - смартфоны, помимо своей мощности приобрели также и другие весьма важные для современного человека характеристики. Это, несомненно, большой экран и камера, позволяющая получать многомегапиксельные изображения окружающего мира. Благодаря камерам в смартфонах и видеокамерам на улицах города, как частным, так и государственным, стремительно возросло количество фотографий и видеоматериалов, которые, в итоге, спустя какое-то время или незамедлительно оказываются в сети Интернет.

В силу всех этих причин, а также принимая во внимание, что количество научных исследований, использующих различную фото- и видеоаппаратуру также возросло, возникает необходимость автоматической обработки подобных данных. Учёным могут потребоваться более современные механизмы для выделения на изображении всех объектов для дальнейшего их сравнения или подсчёта. Обработка может вестись в различных направлениях: из-за постоянной угрозы террористических актов, при известном изображении преступника, государству требуются производительные подходы к распознаванию подозрительных личностей на видеокамерах, например, в аэропорту, и дальнейшему отслеживанию их на видео. Обычным гражданам технологии могут предоставить новые развлечения или более удобные инструменты в случае работы с различными графическими редакторами. Из всех вышеописанных причин следует, что важность алгоритмов в области компьютерного зрения возрастает.

Из сказанного также следует, что необходимо исследовать новые подходы и совершенствовать старые в решении обработки изображений. В частности, необходимо заботиться о скорости и точности выполнения алгоритмов. При постоянно возрастающем объёме требующих проверки данных, скорость будет являться одним из ключевых факторов в выборе алгоритма.

Операция выделения контура является одним из базовых алгоритмических подходов, открывающих возможности для применения алгоритмов использующих более высокую степень абстракции. Таким образом, улучшения возможностей различных подходов к выделению контура благотворно скажется на решении многих задач компьютерного зрения.

Постановка задачи

Рассматривается возможность реализации поиска контура на изображении с применением методологии эволюционных вычислений. Необходимо составить программу, подтверждающую состоятельность такого подхода. В результате программа должна находить и отображать контур единственного объекта на заданном изображении, с возможностью ручной или произвольной инициализации начальных точек.

Решение должно обладать следующими свойствами:

1. Необходимо игнорировать наличие незначительных искажений на контуре;
2. Алгоритм должен предоставлять возможность простой замены функции принадлежности, что может быть использовано при необходимости изменения её работы.

В рамках решения необходимо выполнить следующее:

1. Рассмотреть один из алгоритмов семейства модели активного контура и возможности его модификации для использования в контексте генетического алгоритма;
2. Определить генетическое представление задачи: выбрать, что будет являться генами в задаче, как будут представляться особи, и каким образом будут происходить основные этапы эволюции;
3. Необходимо определить функцию приспособленности, которая позволит найти решение и будет давать адекватную оценку уже найденным решениям.
4. Реализовать генетический алгоритм, используя определенные ранее понятия и протестировать его работу.

Обзор литературы

Модель активного контура под названием Snakes была предложена впервые Kass et al. в 1988 году[1], она описывала представление контура как набора соединенных точек (сплайнов, в русскоязычных источниках часто используется название "змейки"), а также нахождение точного решения для поиска достоверного контура, что требовало значительных ресурсов. Впоследствии данная модель получала различные модификации, одна из которых её реализация в виде динамического алгоритма, что было предложено Williams et al. в их работе[2], описывающей реализацию змеек на основе жадного алгоритма и другие подходы к расчету функционала энергии, что позволило ускорить работу алгоритма.

Несмотря на то, что концепция генетических алгоритмов была известна задолго до появления модели Snakes (Holland, 1975[3]), одними из первых для поиска контура на изображении их предложили применять Rouselle et al. в 2003 году[4], в дальнейшем данный подход совершенствовался, и были предложены его вариации для различных задач: определение контура левого желудочка в сердце (Mishra et al., 2003[5]) или использование вейвлет для предобработки (Mun et al., 2004[6]), что позволило повысить точность распознавания. Работа Mun et al. является более интересной, потому что в ней предлагается задание области интереса для змейки, что ускоряет сходимость алгоритма, но требует большего количества действий от пользователя такого решения.

Работа 2007 года, представленная Ballerini[7], рассматривает различные подходы к реализации генетического алгоритма для поиска контура, обобщая описанные ранее и другие методы.

1 Snakes: active contours model

Рассмотрим предложенную Kass et al. модель активного контура. По своей сути, змейки - это сплайны, которые решают задачу поиска контура путём минимизации функции энергии. Данная модель интересна тем, что в зависимости от определения энергии она применима для широкого круга задач, начиная от определения граней и заканчивая отслеживанием движения. Предложенная Kass et al. модель также позволяет пользователю помочь программе быстрее найти правильное решение за счет ручного выбора начальных точек на изображении или указания направления движения змеек, что способствует быстрой сходимости алгоритма.

Каждый из контуров в данной модели определяется следующими компонентами:

1. Набором точек $v_i, i = \overline{0, n-1}$, задающих координаты змейки на изображении, где n - это количество узлов;
2. Внутренней энергией контура E_{int}, которая контролирует, каким образом змейки могут изменяться:
   (a) Функционал, который определяет равномерность распределения точек змейки;
   (b) Функционал, обеспечивающий гладкость и препятствующий появлению изломов;
3. Внешней энергией контура E_{ext}, которая обычно представляет комбинацию сил на изображении E_{image} и ограничений E_{const}, задаваемых пользователем. Данная энергия и будет направлять змейки к контуру, являющимся решением задачи.

Таким образом, функция энергии змейки, которую она должна минимизировать для нахождения контура объекта имеет вид:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds. \quad (1)$$

1.1 Внутренняя энергия

Внутреннюю энергию можно определить следующим уравнением:

$$E_{int} = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)/2, \tag{2}$$

где первое слагаемое контролирует, чтобы в кривой не было больших различий между расстояниями в контуре, второе необходимо для сохранения гладкости контура. Значения α, β задаются пользователем в зависимости от задачи.

1.2 Энергия изображения

Энергия изображения определяется следующим образом:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term},$$

где $w_{line}, w_{edge}, w_{term}$ задаются пользователем в зависимости от задачи.

E_{line} в простейшем случае может определяться значением интенсивности изображения. Таким образом, если установить данную энергию равной значению интенсивности изображения или этому значению, взятому с обратным знаком, то змейки будут притягиваться к светлым или темным местам на изображении.

Данный функционал также может задаваться функцией от изображения, если положить $E_{edge} = -|\nabla I(x, y)|^2$, то змейки будут притягиваться к контурам с сильным градиентом.

Последний функционал E_{term}, определяющий был ли найден контур можно определять при помощи кривизны.

1.3 Ограничения пользователя

Функционал E_{const} оставляет возможность пользователю самому направлять змейки в направлении поиска контура. Это может быть интерактивное направление путём воздействия с программой, реализующей алгоритм в реальном времени, основанное на некоторых характеристиках (features) изображения или путем задания направления движения змеек.

2 Генетический алгоритм

Генетический алгоритм (ГА) - один из видов эволюционных вычислений, который применим для различных задач оптимизации с определённой заранее целевой функцией, в рамках алгоритма называемой функцией приспособленности. Впервые данный подход предложен Холландом в 1975 году[3]. Такое название алгоритма объясняется тем, что он во время решения задачи имитирует эволюционные процессы: используются понятия генов, особей, хранящих наборы этих генов, популяций, объединяющих группы особей. При этом для получения новых решений используются операции мутации, кроссинговера и отбор наиболее приспособленных решений.

В контексте генетического алгоритма важно понимать различие между генотипом и фенотипом:

1. Генотип - совокупность генов данного организма;
2. Генофонд - совокупность всех возможных вариаций генотипов;
3. Фенотип - отражение генотипа в реальность, то есть набор признаков, которые обретает организм при том или ином наборе генов.

Типичные операции, используемые в генетическом алгоритме:

1. Мутация - с некоторой заранее определённой вероятностью произвольное изменение любого гена или нескольких в геноме одной особи;
2. Кроссовер - операция получения нового генома, происходящая путём смешения генов двух и более родителей;
3. Отбор - некоторое правило выбора родителей для совершения описанных выше операций;
4. Расчёт функции приспособленности - необходим для проведения оценки особей, на основе данного значения выполняется операция отбора.

При использовании генетического алгоритма решение задачи будет получаться путем многократного применения указанных выше операций к каждой

возникающей популяции. Каждая новая популяция должна иметь значение функции приспособленности для наиболее приспособленных особей не хуже, чем предыдущая популяция. Обычно алгоритм останавливается по достижению порогового числа популяций. Доказательство конечности данного алгоритма приведено Холландом в его работе, где впервые были представлены генетические алгоритмы [3], и названо теоремой схем Холланда.

3 Возможность применения генетического алгоритма

3.1 Жадные змейки

В качестве отправной точки выбран алгоритм жадных змеек, ввиду простоты для реализации и хорошего баланса скорость-качество. Данный алгоритм является модификацией оригинальной модели активного контура, предложенный Donna et al. в 1991 году[2].

Данный алгоритм основан на том, что новое местоположение для точки змейки на каждом этапе определяется из правила локальной оптимальности, выбором лучшего места из ограниченного набора соседних точек. Именно благодаря этому алгоритм и получил название "жадного".

Другое нововведение, сделанное в работе Donna et al. - более точный способ расчета кривизны. В их варианте змейки должны были также минимизировать энергию по форме уравнения (1), но со следующими различиями в определении функционалов:

1. E_{image} имеет такой же смысл как и в исходном уравнении;

2. E_{cont} определяется как:

$$\bar{d} - |v_i - v_{i-1}|,$$

   где $\bar{d}$ - средняя дистанция между последовательными парами точек в змейке. Также предлагается нормализовать данный терм делением его на максимальное значение, достигаемое для соседей данной точки. Это будет заставлять змейки стремиться к тому, чтобы между всеми их узлами было одинаковое расстояние, что будет приводить к равномерному контуру;

3. E_{curv}, определяющий кривизну сегмента, предлагается рассчитывать по формуле:

$$|v_{i-1} - 2v_i + v_{i+1}|^2,$$

данную формулу предлагается применять для нахождения лучшего нового места для точки змейки. Но для оценки близости к контуру предлагается использовать более точную и вычислительно сложную формулу:

$$|\frac{\vec{u_i}}{|\vec{u_i}|} - \frac{\vec{u_{i+_1}}}{|\vec{u_{i+1}}|}|.$$

3.2 Реализация

Данный алгоритм реализован в среде Matlab v2013b по псевдокоду, изложенному в работе Donna et al. [2], в качестве демонстрации работы подхода и определения плюсов и минусов.

Как можно видеть на представленных изображениях (рисунок 1), данная модель сильно зависит от дальности начального расположения точек от искомого контура. Чем дальше находятся начальные точки от контура, тем меньше шансов, что алгоритм не попадёт в локальный минимум, из которого уже не сможет выбраться.

3.3 Возможность применения генетического алгоритма

В рассмотренном алгоритме и других алгоритмах семейства Active Contours Model для нахождения контура необходимо минимизировать функцию энергии. В связи с этим функция энергии может выступать в роли функции приспособленности для генетического алгоритма. В качестве генов можно взять точки, особями же будут сами змейки. В целом, можно отметить, что модель активного контура допускает естественную интерпретацию в контексте генетического алгоритма. Основное отличие от рассмотренных работ будет заключаться в том, как именно будет получаться каждая новая змейка, и, как следствие, каким образом будет осуществляться поиск минимума функционала энергии.

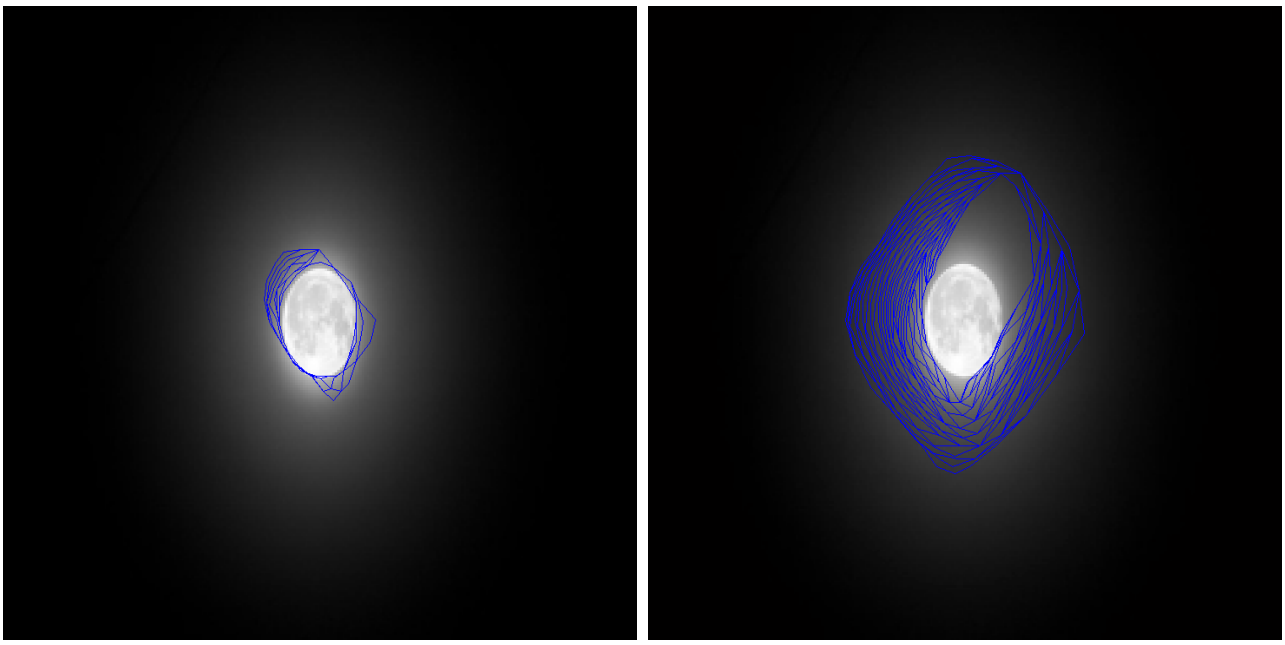


Рис. 1: Примеры выполнения жадных змеек

4 Описание алгоритма

4.1 Генетический алгоритм

Для решения поставленной задачи необходимо переопределить стандартные понятия и операции генетического алгоритма, чтобы его можно было применять для решения задачи поиска контура.

Как описано выше, в качестве генов будут взяты точки. Тогда генотипом будет множество всех возможных точек на изображении. Определим необходимые понятия для генетического алгоритма:

4.1.1 Особь

Исходя из того, что точки являются генами, то наборам генов, то есть особями, будут являться змейки: они представляются как наборы точек и, соответственно, подходят на эту роль. Змейки также будут характеризоваться набором компонент, схожим с тем, что имеют контуры в Active Contours

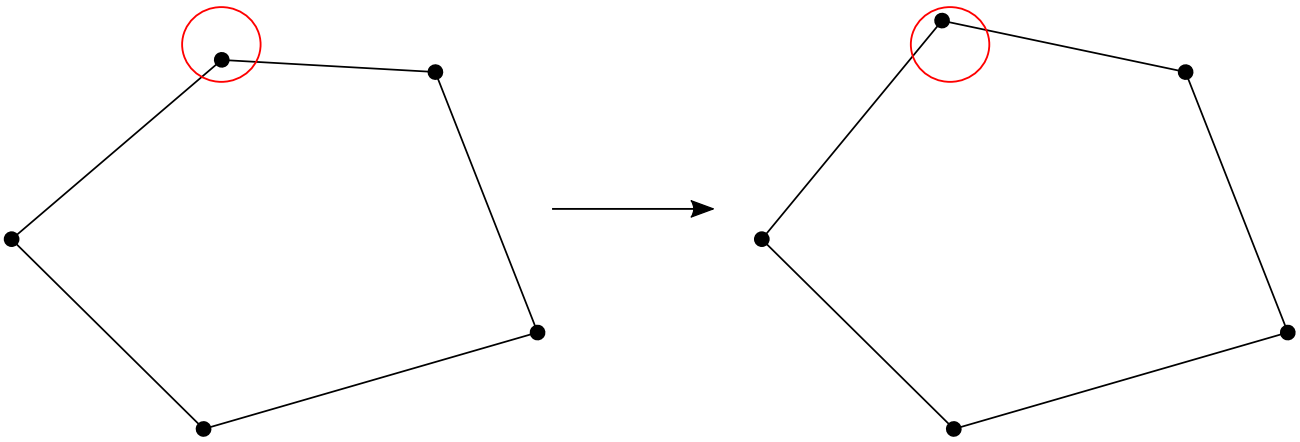


Рис. 2: Пример мутации контура

Model:

1. Набором генов $g_i, i = \overline{0, n-1}$, представляющим собой набор точек, отображение которых на изображении проявляет фенотип змейки - её расположение на изображении;
2. Значением функции приспособленности, подробнее о которой будет сказано ниже.

4.1.2 Популяция

Популяция логичным образом определяется как набор змеек.

4.1.3 Мутация

Логично ввести операцию мутации следующим образом - для точки, подверженной мутации, будет выбираться другая точка из заранее заданной окрестности, если новая точка находится на изображении, то исходная точка заменяется на найденную (рис. 2). В случае, когда найденная точка не находится в пределах изображения, ищется новая точка.

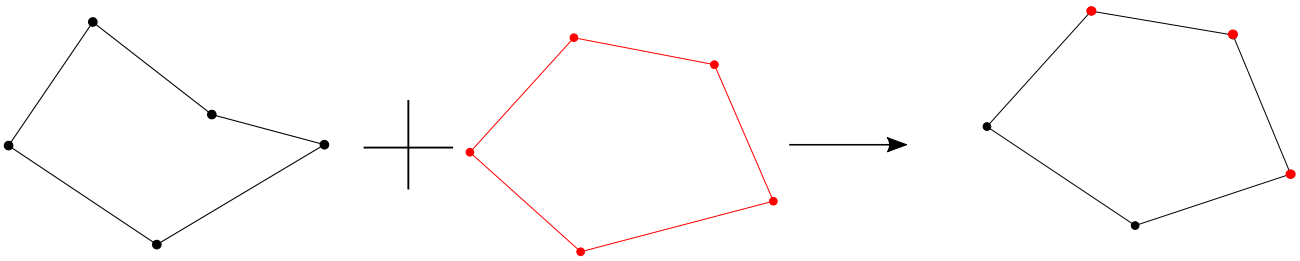


Рис. 3: Пример выполнения одноточечного кроссовера

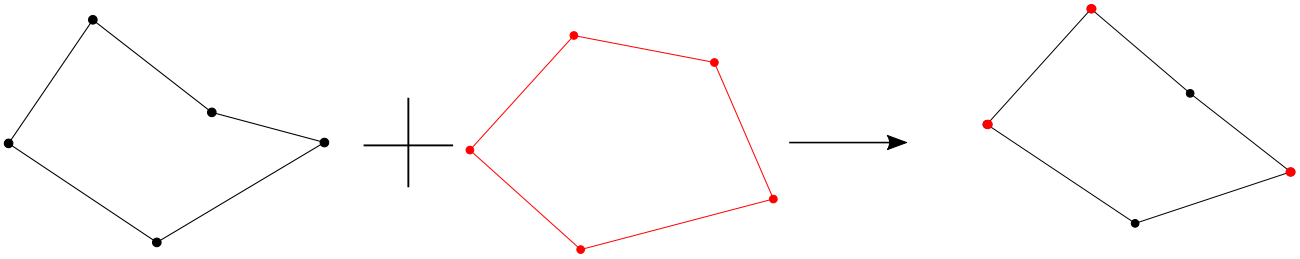


Рис. 4: Пример выполнения "однородного" кроссовера

4.1.4 Кроссовер

Реализованы два вида кроссовера:

1. Оба родителя разрезаются в некоторой произвольной точке, затем части одного родителя соединяются с частями другого и получается новая особь. Такой кроссовер имеет название одноточечного (one-point crossover), также его иногда называют кроссовером на основе разрезания и соединения (Cut and Splice) (рис. 3);

2. Единственному потомку с некоторой вероятностью присваиваются гены от одного или другого родителя. Данный вид кроссовера называют "однородным"(uniform) (рис. 4). Он примечателен тем, что, по своей сути, он моделирует действие всех возможных типов кроссовера на основе разрезания и соединения, предоставляя гораздо большее количество вариаций [8].

4.1.5 Отбор

Был реализован отбор турниром. Идея данного метода заключается в том, что для выбора каждого кандидата проводится несколько "турниров"(от 1 и более), на каждом из которых происходит выбор случайным образом одной особей популяции, чтобы затем сравнить её приспособленность с той, что была выбрана до этого. Отбор турниром реализован ввиду его простоты, а также возможности легко влиять на обучение путем изменения количества турниров. Можно отметить, что при количестве турниров равным 1, данный вариант отбора ничем не отличается от случайного выбора одной особи из популяции.

4.1.6 Функция приспособленности

В качестве функции приспособленности использован измененный функционал энергии (1), предложенный в работе Kass et al. Необходимо отметить, что исходное изображение приводится к типу в градациях серого, после чего применяется сглаживание фильтром Гаусса.

Основные термы определены следующим образом:

1. В качестве энергии изображения используется значение пикселя изображения после применения фильтра Лапласа:

$$E_{image} = \sum_{0}^{n} G_\sigma * \nabla I(v_i),$$

   здесь и далее для первой точки в качестве предыдущей берётся последняя, для последней в качестве последующей - первая;

2. В качестве внутренней энергии E_{int} используется сумма следующих компонент:

(a) Сумма значений кривизны каждого сегмента контура:

$$E_{curv} = \sum_{0}^{n} |v_{i-1} - 2v_i + v_{i+1}|^2,$$

имеет смысл второго слагаемого в формуле (2);

(b) Сумма отклонений расстояния между точками от среднего расстояния между точками:

$$E_{avg} = \sum_{0}^{n} (\bar{d} - |v_i - v_{i-1}|),$$

имеет смысл первого слагаемого в формуле (2);

(c) Площадь покрываемая контуром:

$$E_{area} = \frac{1}{2} \sum_{0}^{n} (v_i.x * v_{i+1}.y - v_{i+1}.x * v_i.y),$$

введение данного терма заставляет змейки минимизировать свою площадь, что стимулирует алгоритм к более быстрому нахождению контура;

(d) Штраф за то, что точки находятся слишком близко друг от друга, данный терм необходим, так как минимизируя свою площадь и энергию изображения, контур может сходиться в одну точку, имеющую максимальное значение энергии изображения:

$$E_{points} = \begin{cases} \Omega, & \text{if } \exists i, j : i \neq j \& distance(v[i], v[j]) < \sigma \\ 0, & \text{иначе} \end{cases},$$

где Ω - некоторое большое значение, представляющее собой штраф, позволяющий отсеивать такие змейки из будущих популяций, σ - расстояние задаваемое пользователем, ближе которого точки контура не должны находиться.

3. Терм E_{const}, представляющий ограничения пользователя на изображении, не использован и принят равным нулю.

Таким образом, исходную задачу можно сформулировать в виде оптимизационной задачи:

$$\alpha E_{image} + \beta E_{curv} + \gamma E_{avg} + \delta E_{area} + \theta E_{points} \to \min_{\vec{v} \in \mathrm{V}}, \tag{3}$$

где V представляет собой генофонд - множество всех точек изображения, а $\alpha, \beta, \gamma, \delta, \theta$ - константы, задаваемые пользователем.

4.2 Алгоритм

Для удобства восприятия псевдокод алгоритма разбит на несколько частей. Основной цикл представлен на листинге 1. В данной реализации предполагается, что пользователь алгоритма может определить свою собственную функцию, удовлетворяющую предложенному интерфейсу и использовать её в качестве функции приспособленности (соответственно, все определяемые пользователем константы задаются на этапе инициализации такой функции).

Рассмотрим подробно, что именно скрывается за вызовом метода `EvolvePopulation` (листинг 2). После расчета значения функции приспособленности для каждой змейки они сортируются по убыванию приспособленности. Это необходимо для того, чтобы на следующем этапе отобрать `ElitAmount` змеек, которые войдут в новую популяцию без изменений. Данный шаг приводит к тому, что наиболее приспособленные особи из текущей популяции переносятся в следующую, что даёт более качественное потомство и, соответственно, ускоряет сходимость (доказательство приведено в работе Yang [9]). Все остальные места в новой популяции (её размер совпадает с исходной) заполняются при помощи операторов скрещивания, действие которых было описано выше. После этого с шансом `MutateRate` происходит мутация всех змеек, кроме тех, что были отобраны как неизменяемые.

Листинг № 1: Главный цикл

```
Input: fitnessFunction, N
Output: contour
population = initSnakesRandomly();
for i = 0 to N do
    population = population.EvolvePopulation(fitnessFunction);
end
fittest = population.GetFittest();
return fittest;
```

Листинг № 2: Эволюция популяции

```
Input: population, fitnessFunction
Output: newPopulation
population.CalcFitness(fitnessFunction);
population.SortDescending();
newPopulation[0, ElitAmount-1] = population.GetFirstN(ElitAmount);
for i = ElitAmount to PopulationSize do
    parentA = TournamentSelection();
    parentB = TournamentSelection();
    if Random() < UCrososverRate then
        child = UniformCrossover(parentA, parentB);
    end
    else
        child = Cut&SpliceCrossover(parentA, parentB);
    end
    newPopulation[i] = child;
end
for i = ElitAmount to populationSize do
    if Random() < MutateRate then
        Mutate(newPopulation[i]);
    end
end
return newPopulation
```

5 Реализация алгоритма

5.1 Реализация

Алгоритм поиска контура реализован на языке C# версии 6 в среде Microsoft Visual Studio 2015. C# был выбран как современный высокоуровневый язык, который позволяет писать производительный код с сохранением высокой скорости разработки. Язык высокого уровня предпочтительнее для реализации генетического алгоритма, так как принцип популяций, состоящих из особей (а особи состоят из генов), хорошо ложится на объектно ориентированную парадигму программирования.

Помимо стандартных библиотек, которые поставляются со средой разработки, использованы следующие:

1. Библиотека Emgu CV версии 3.1.0.2282[10], которая использована для упрощения работы с изображениями: при помощи неё реализованы операции загрузки изображения, его отображения, а также некоторые алгоритмы над изображениями, такие как Гауссово сглаживание или фильтр Лапласа, которые использованы для предобработки изображения в функции приспособленности;
2. Библиотека MoreLinq[11], позволившая упростить написание кода, работающего с последовательностями элементов.

Код алгоритма оформлен в виде библиотеки, которая предоставляет методы и типы, позволяющие решать поставленную задачу поиска контура на изображении. Требование пункта 2 раздела постановки задачи было решено следующим образом: библиотека предоставляет абстрактный класс, при реализации которого любой класс, определённый пользователем данной библиотеки, можно будет использовать в качестве функции приспособленности.

При необходимости предусмотрена возможность изменить параметры, используемые алгоритмом для работы, для этого библиотека предоставляет публичные свойства. Изменениям могут быть подвержены многие параметры,

включая следующие:

1. Количество точек, составляющих змейку;
2. Вероятность выполнения кроссовера;
3. Вероятность мутации змейки;
4. Количество змеек отбираемых в элиту;
5. Максимальная дистанция от текущей точки, из которой ведётся поиск новой точки во время мутации.

В данной реализации предполагается, что все контуры имеют одинаковую, заранее фиксированную длину. Существуют два разных подхода к созданию начальной популяции:

1. Пользователь может задать некоторый начальный контур, от которого путем проведения мутаций будут образованы все остальные контуры первой популяции;
2. Начальная популяция может быть полностью сгенерирована случайным образом.

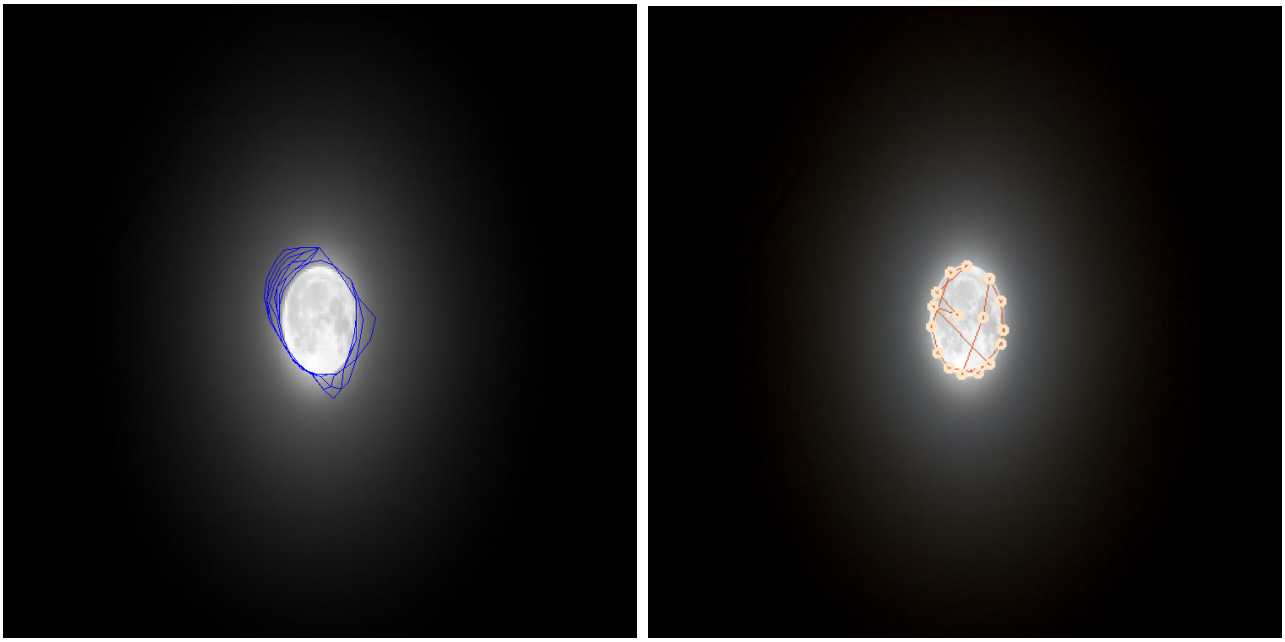
Ввиду особенностей выполнения генетического алгоритма конечный контур может быть получен верно, но с некоторым количеством самопересечений. В таком случае, к полученному контуру можно применить функцию, которая позволит переупорядочить точки в змейке. Один из возможных вариантов реализации такой функции:

1. Нахождение центроидной точки контура;
2. Перевод координат точек контура в полярные относительно найденной на предыдущем этапе точки;
3. Сортировка точек в полярных координатах по величине угла;
4. Обратный переход от полярных координат к координатам на изображении.

5.2 Тестирование

В процессе тестирования опытным путём установлены следующие настройки алгоритма, позволяющие найти контур:

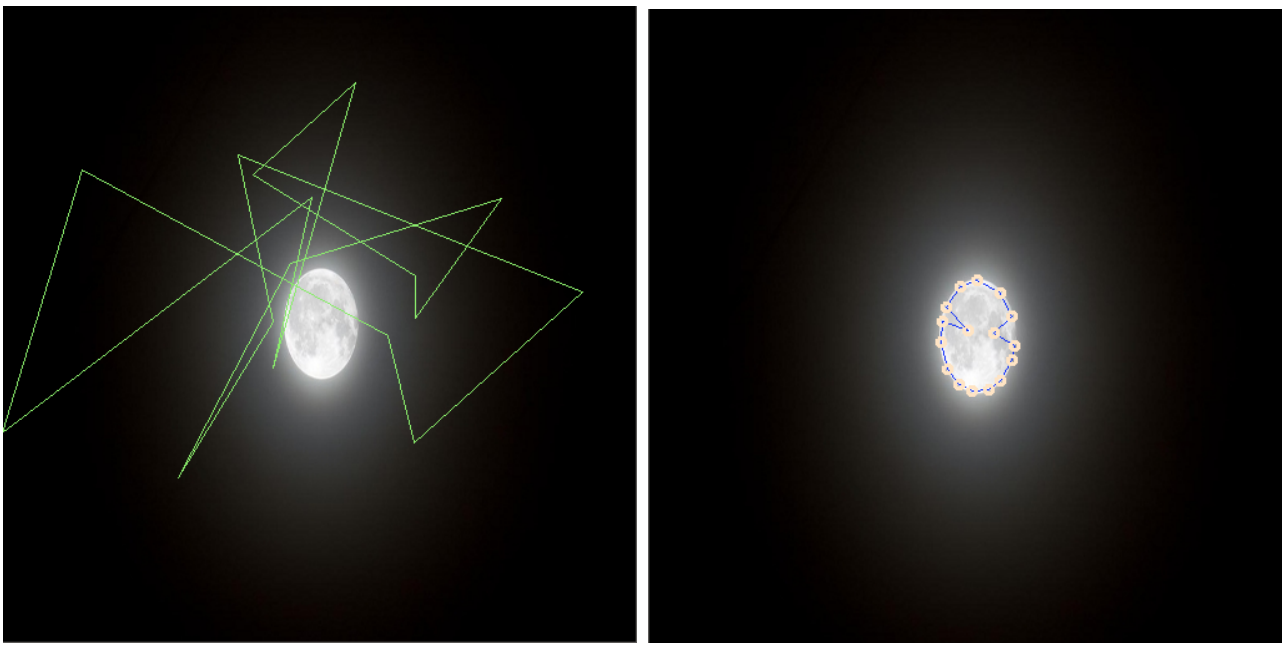
1. Используются особи из 16 точек, что позволяет сохранить баланс между точностью и скоростью выполнения;
2. В популяции количество особей равно 100, что также обусловлено точностью и скоростью выполнения алгоритма;
3. Для уравнения (3) подобраны следующие значения коэффициентов $\alpha = 0.1, \beta = 9, \gamma = 1, \delta = 0.05, \theta = 1$



(a) Жадные змейки (b) Генетический алгоритм

Рис. 5: Сравнение результатов работы генетического алгоритма и реализованного ранее поиска контура жадными змейками

Как показано на рисунке 5, генетический алгоритм позволяет найти контур не хуже, чем это было сделано при помощи жадных змеек. Необходимо отметить, что жадные змейки были инициализированы вручную в непосредственной близости от искомого контура, генетический алгоритм был инициализирован случайными точками (изображение 6a).



(a) Начальные точки (b) Пример исправленного контура

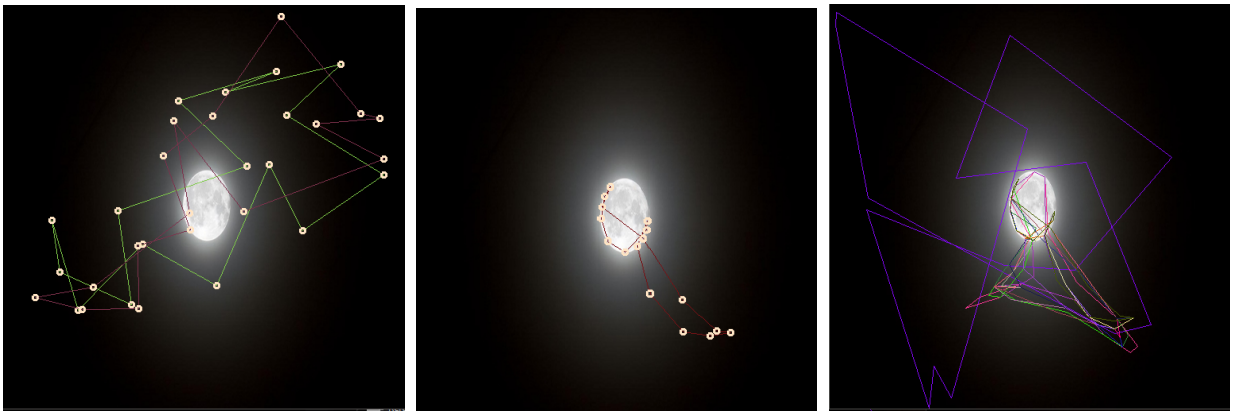
Рис. 6: Результат работы генетического алгоритма

На изображении 5b видно, что контур получился с самопересечениями, для устранения этого недостатка, к нему была применена описанная ранее функция, переопределяющая порядок точек. В итоге получен исправленный контур представленный на рисунке 6b.

В алгоритме использовано значение `ElitAmount` $= 5$, подобранное опытным путем. Заметим, что при других значениях параметра выявлены следующие проблемы:

1. Из-за того, что накопленные изменения теряются практически на каждом шаге (шанс мутации - 0.5), ни один контур после всех популяций не описывает объект на изображении (рисунок 7a);

2. При увеличении данного параметра накопленные изменения не будут потеряны и контур частично будет найден (рисунок 7b), но, проанализировав лучшие решения, становится видно, что алгоритм попал в локальный минимум, из которого не мог выбраться в силу малого разнообразия потомства (рисунок 7c).



(a) `ElitAmount` = 0 и показаны два лучших решения

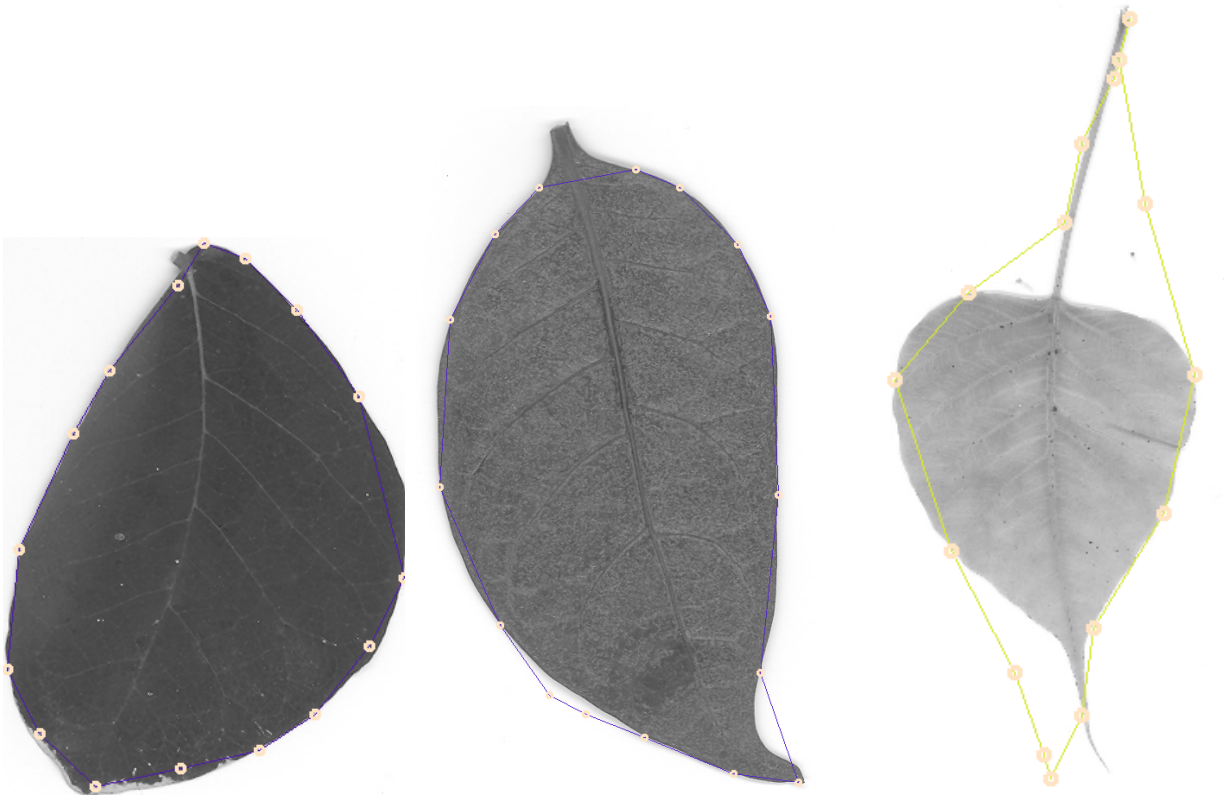
(b) `ElitAmount` = 10

(c) `ElitAmount` = 10 и показано лучшее решение каждой тысячной популяции. Использовано 30 тысяч популяций

Рис. 7: Различные настройки значения `ElitAmount`

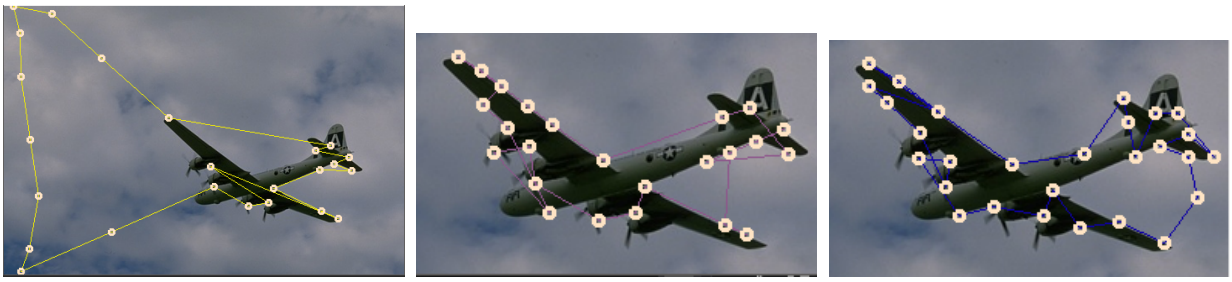
Для дальнейших тестов использовался набор данных предоставленный университетом Беркли[12] и Leaf Image Database[13].

Тестирование на Leaf Image Database (рисунок 8) позволяет показать, что скорость схождения алгоритма напрямую зависит от того, насколько начальный контур близок к искомому. Хорошие результаты получены всего после прошествия 1000 популяций. Необходимо отметить, что для данных изображений начальной позицией контура являлись границы самого изображения, что обусловлено тем, что объект занимает практически всё изображение, а терм площади E_{area} не будет допускать рост области занимаемой змейками. Также на рисунке 8c видно, что даже после 10 тысяч популяций змейки не смогли достаточно приблизиться к контуру, что вызвано тем, что данное изображение имеет сильно вытянутый объект, с утолщением, охватить который сложно при выполнении ограничения на гладкость контура и малом количестве точек.



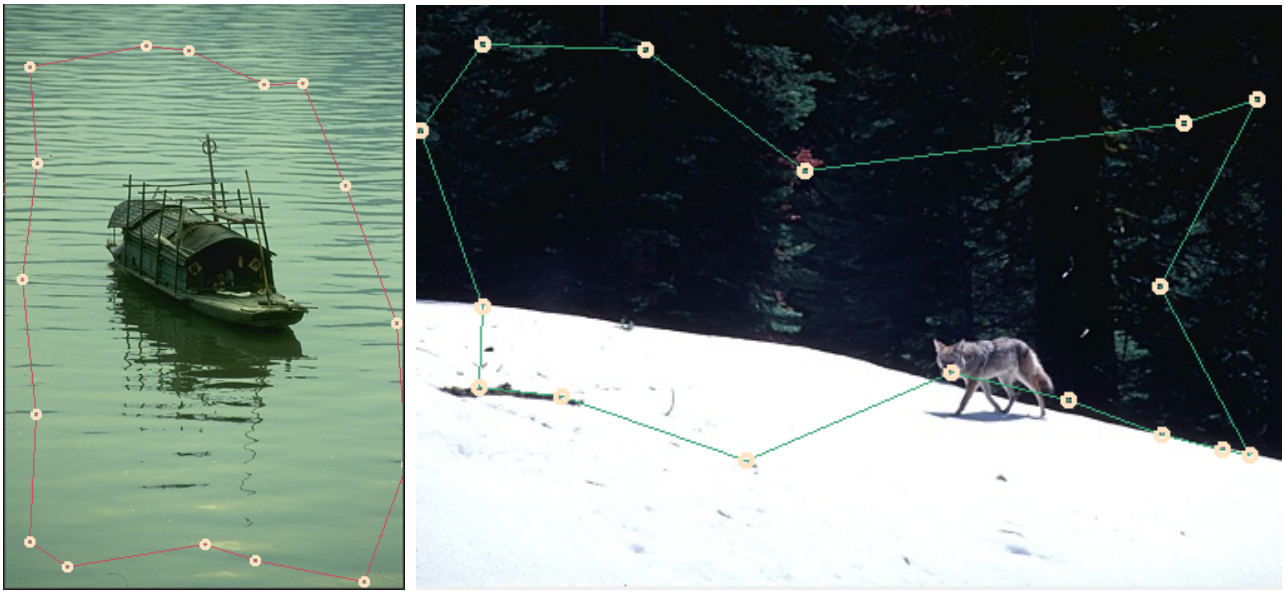
(a) 1000 популяций (b) 1000 популяций (c) 10 тысяч популяций

Рис. 8: Разные примеры выполнения на изображениях из набора Leaf Image



(a) 50 тысяч популяций (b) 10 тысяч популяций и уменьшена дальность мутации точек (c) 10 тысяч популяций, но большее количество точек

Рис. 9: Разные примеры выполнения в зависимости от количества точек и начальной позиции



(a) 30 тысяч популяций (b) 10 тысяч популяций

Рис. 10: Примеры неправильных решений

Изображения, входящие в состав набора изображений Беркли являются более сложными для определения контура, так как это реальные снимки, обладающие естественным неоднородным фоном.

На рисунке 9 можно наблюдать как начальная инициализация на изображениях может повлиять на точность определения контура. Например, на рисунке 9a после большого количества популяций контур найден частично, часть точек осталась в локальном минимуме. Однако, если начать поиск контура в непосредственной близости от самолёта с уменьшением максимальной дальности мутации точек, то после всего 10 тысяч популяций алгоритм найдёт неплохое решение (рисунок 9b), которое можно улучшить, если увеличить количество точек в змейках (рисунок 9c).

На рисунке 10 можно наблюдать примеры выполнения алгоритма, не приведшие к искомому контура. В случае изображения 10a этому препятствовали блики на воде, представляющие для алгоритма точки локальной оптимальности, так как предоставляют большое значение терма E_{image}. Таким образом после того как змейки на первых итерациях попали в эти точки, ни один из потомков 30 тысяч популяций не смог предоставить лучшего значения функции приспособленности. Для нахождения контура на таком изображении требует-

ся дополнительная предобработка изображения, позволяющая убрать подобные точки локальной оптимальности.

На изображении 10b поиску правильного решения также помешали точки локальной оптимальности, кроме того, оказало свою роль наличие на изображении границы светлого и тёмного, что алгоритмом воспринимается как граница контура. Поэтому можно наблюдать часть точек, оказавшуюся вдоль этой границы. Обеспечить правильное нахождение контура на данном изображении возможно при начальной инициализации вблизи искомого контура.

Другие примеры результатов работы алгоритма можно найти в приложении.

Выводы

1. Рассмотрен и реализован алгоритм жадных змеек, относящийся к семейству алгоритмов модели активного контура. Показано, что использование генетического алгоритма для поиска контура на изображении целесообразно и предложены определения основных понятий генетического алгоритма в рамках задачи поиска контура;

2. Предложены и описаны возможные реализации основных операций, выполняющихся в рамках генетического алгоритма, в приложении к задаче поиска контура на изображении. Также сформулировано представление особей.

3. Определена и описана функция приспособленности, основанная на функционале энергии, изначально представленным Donna et al. для модели greedy snakes;

4. На языке C# реализован генетический алгоритм поиска контура с понятиями и операциями описанными в данной работе;

5. Проведено тестирования реализованного алгоритма, в ходе которого выявлены следующие проблемы:

   (a) При наличии на изображении точек локальной оптимальности алгоритм может не достигнуть искомого контура. Для решение данной проблемы можно применить дополнительную предобработку изображения или инициализировать контур в непосредственной близости от объекта;

   (b) Из-за стохастичности процесса и большого количества возможных расположений змейки на изображении может потребоваться большое количество времени для нахождения контура. Эта проблема решается уменьшением размера изображения или заданием пользователем начального контура;

   (c) По схожим причинам может потребоваться обработка итогового

контура: в случае наличия самопересечений или гладкого контура с небольшим количество далеко отстоящих точек. Возможным путём решения проблемы самопересечений является введение ещё одного терма, аналогичного терму E_{points}, который будет добавлять штрафное значение к значению функции приспособленности змеек с таким дефектом.

Заключение

В рамках проведённого исследования рассмотрен и реализован один из алгоритмов семейства модели активного контура. На основе него определена возможность применения генетического алгоритма к данной задаче. Применительно к задаче поиска контура определены основные понятия генетического алгоритма, а именно гены, особи, поколения. Определены операции генетического алгоритма, позволяющие получать новых особей: два вида кроссовера, отбор турниром, мутация. В том числе описана функция приспособленности. На основе этих знаний описан и реализован генетический алгоритма поиска контура на изображении.

В связи с этим можно выделить следующие направления дальнейшей работы:

1. Реализовать адаптивный метод установки значений вероятностей кроссовера и мутации (данный подход описан в работе Srinivas et al. [14]);
2. Исследовать возможность модификации данного алгоритма для определения контуров нескольких объектов на одном изображении;
3. Исследовать различные виды предобработки изображения, позволяющие избежать наличия точек локальной оптимальности.

Список литературы

[1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321, January 1988.

[2] Donna J Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image understanding*, 55(1):14–26, 1992.

[3] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[4] Jean-Jacques Rousselle, Nicole Vincent, and Nicolas Verbeke. Genetic algorithm to set active contour. In *Computer Analysis of Images and Patterns*. Springer, January 2003.

[5] A Mishra, PK Dutta, and MK Ghosh. A ga based approach for boundary detection of left ventricle with echocardiographic image sequences. *Image and vision Computing*, 21(11):967–976, 2003.

[6] Kyeong-Jun Mun, Hyeon Tae Kang, Hwa-Seok Lee, Yoo-Sool Yoon, Chang-Moon Lee, and June Ho Park. Active contour model based object contour detection using genetic algorithm with wavelet based image preprocessing. *International Journal of Control Automation and Systems*, 2:100–106, 2004.

[7] L Ballerini. Genetic snakes: Active contour models by genetic algorithms. *Genetic and evolutionary computation in image processing and computer vision, EURASIP book series on SP & C*, pages 177–194, 2007.

[8] Kenneth A De Jong. *Evolutionary computation: a unified approach.* MIT press, pages 64-65, 2006.

[9] Shengxiang Yang. Genetic algorithms with elitism-based immigrants for changing optimization problems. In *Applications of Evolutionary Computing*, pages 627–636. Springer, 2007.

[10] Emgu cv is a cross platform .net wrapper to the opencv image processing library. http://www.emgu.com/wiki/index.php/Main_Page. Accessed: 2016-05-12.

[11] Morelinq extensions to linq to objects. https://morelinq.github.io/. Accessed: 2016-05-12.

[12] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[13] Vaibhav E Waghmare. Leaf image database.

[14] Mandavilli Srinivas and Lalit M Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656–667, 1994.

Приложение

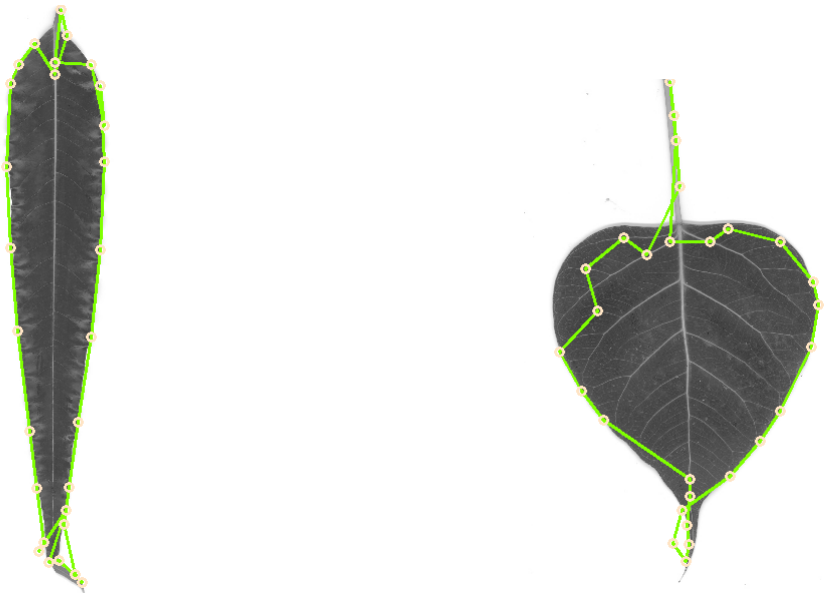
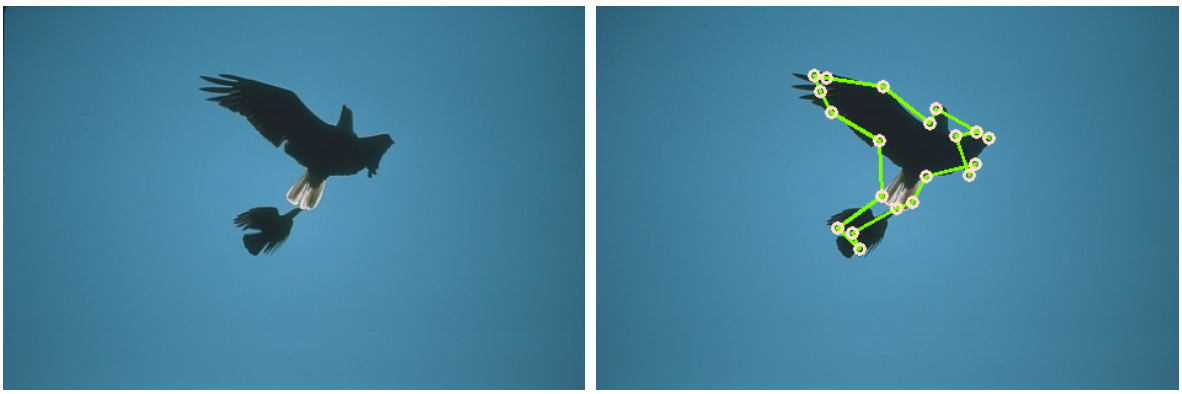


Рис. 11: Примеры выполнения на изображениях из набора Leaf Image

На рисунке 11 можно наблюдать, что алгоритм нашёл решение, но в обоих случаях контур имеет искажения за счет наличия на листочке точек локальной оптимальности. Правый рисунок иллюстрирует утверждение, о том, что на изображении 8c возможно точнее определить контур при увеличении количества точек в змейке.

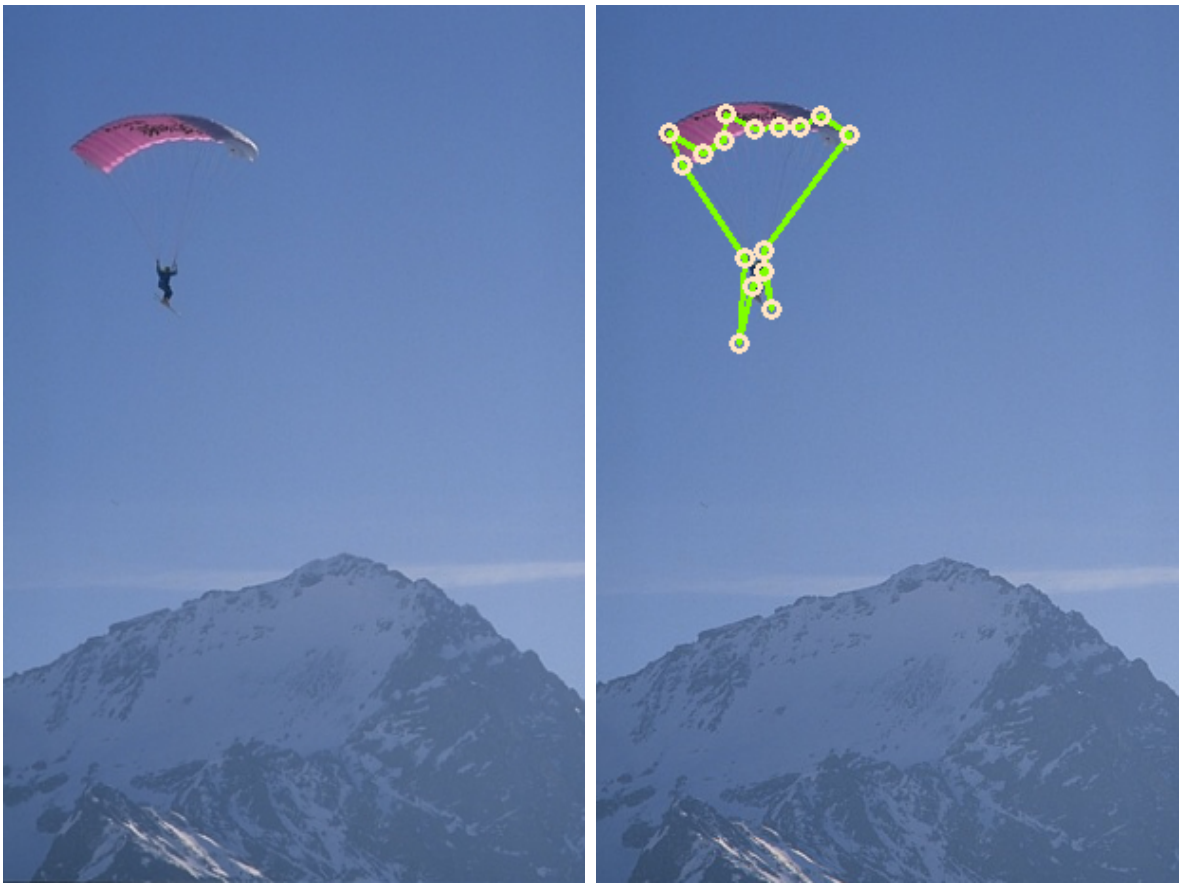


(a) Оригинал (b) Решение после 20 тысяч популяций

Рис. 12: Пример выполнения на изображении из набора Беркли

На рисунках 12 и 13 можно наблюдать, что алгоритм позволяет найти достаточно точное решения для таких сложных объектов. Для достижения

необходимой точности уменьшен параметр, отвечающий за дальность мутации точки. Уменьшение данного параметра, также требует увеличения количества популяций, по причине того, что змейки, подвергшиеся мутации, будут расположены ближе к своему положению до мутации, что обуславливает более медленную сходимость алгоритма.



(a) Оригинал (b) Решение после 20 тысяч популяций

Рис. 13: Пример выполнения на изображении из набора Беркли